MiniSQL 报告

计算机科学 徐延成 3160101851

一、功能概述

miniSQL 的整体设计要求数据库有建立，删除表和索引，插入、删除、搜索数据。表中支持 int，float 和 char 三种数据类型，表的属性可以定义为其中任意的数据类型，同时支持将属性设定为 unique 或是表的主键。对于定义为 unique 的属性可以生成 B+树索引。

二、程序分块

程序分为表，索引，数据库管理和命令解析几个方面。

表类在其中存储了自身属性的具体信息（名称，数据类型，是否 unique 等）和内容数据，并提供了读写文件，和对表内部数据搜索，以及表输出的方法。

索引内部有指向表的指针，为其中的 unique 的属性建立 b+树索引，和通过索引寻找数据的方法。通过 b+树可以快速找到相应的数据，并打包返回。

数据库管理系统部分综合了表和索引两部分，提供了两部分沟通的功能。同时他也提供了整个数据库的 API，可以通过它提供的函数来操作整个数据库。

命令解析可以解析输入的 SQL 命令，并把它转化为 API 可以识别的形式

三、API 设计

1） 创建索引

dbCreateIndex(string indexName, string tableName, int attrNo);

2） 创建表

dbCreateTable(string tableattr);

3） 相等值查找

dbSearch(string tableName, int attrNo, string Value);

4） 不等查找

dbSearch(string tableName, int attrNo, float lowerValue, float higherValue, bool equal1, bool equal2);

5） 删除表

dbDropTable(string tableName);

6） 删除索引

dbDropIndex(string tableName, int attrNo);

dbDropIndex(string indexName);

四、关键算法

1） B+树插入

```
1.  BPlusNode* BPlusNode::recursiveInsert(BPlusNode* T, KeyType keyValue, int tPos,
    BPlusNode* Parent) {
2.
3.   while (j < T->keyNum && keyValue >= T->key[j]) {
4.       if (是重复值)
5.           return T;
6.       j++;
```

```
    }
    if (j != 0 && T->children[0] != NULL)
        j--;

    if (T是叶节点) {
        T中插入Key
        if (T有父节点)
            Parent->key[tPos] = T->key[0];
    }
    else
        T->children[j] = recursiveInsert(T->children[j], keyValue, j, T);
    Limit = M;
    if (T中元素数目超过M) {//如果需要分裂节点
        if (T没有父节点)
            分裂T节点;
        else {
            Sibling = T相邻的儿子没有满的节点;
            if (Sibling != NULL) {
                将T一个元素移动到Sibling中;
            }
            else {
                将T分裂为两个节点;
            }
        }
    }
    if (T有父节点)
        Parent->key[tPos] = T->key[0];
    return T;
}
```

2）Table 插入

```
void Table::tableInsert(string toAdd) {
    for (int i = 0; i < MAX_ATTR_NUM; i++) {
        if (attrName[i] == "") break;
        if (unique[i] == true) {
            检查待插入的记录中所有的属性是否和表中unique属性没有重复
            if (没有重复)
                将记录插入;
        }
    }
    BufferNode* focus = next;
    if (buffer无内容) {
        创建新的bufferNode;
```

```
}
while (focus != NULL && focus->next != NULL) {
    if (这个BufferNode没有满)
        break;
    focus = focus->next;
}
if (focus != NULL && focus->recordNum < BUFFER_CAPACITY ) {
    将记录查到bufferNode的最后;
    focus->recordNum++;
    focus->length += toAdd.length();
}
else if (没有可以插入的位置) {
    创建新的节点并插入;
}
return;
}
```

3) Table 搜索

```
Table Table::searchTable(int attrNo, double lower, double higher, bool equal1, bool equal2) {
 对应this table创建一张新的表
 BufferNode* focus = head.next;
 while (focus != NULL) {
     for (i = 0; i < focus中的记录数; i++) {
         attrvalue = focus中的第i条记录中的相应属性;
         根据equal1, equal2的值确定是否符合条件
         if(符合条件)
            将这一条记录插入result;
     }
     focus = focus->next;
 }
 return result;
}
```

4) Table 选择

```
Table Table::selectTable(vector<int> attrNo) {
根据选择的属性创建新的表result;
 for (i = 0; i < 原表的总记录数; i++) {
     newcontent = 提取原表中第i个记录中的需要属性;
     将newcontent插入result中;
 }
 return result
}
```